SOLID

Programação Orientada a Objeto II

Princípios SOLID

SOLID é um acrônimo para cinco princípios de design a fim de orientar um código mais compreensível, flexível e com fácil manutenção.

Os cinco princípios propostos por *Robert C. Martin* em seu livro **Código Limpo**, são:

S - Single-Responsibility Principle
O - Open-Closed Principle
L - Liskov Substitution Principle
I - Interface Segregation Principle
D - Dependency Inversion Principle

Background histórico

Frequentemente, os princípios do SOLID são associados ao movimento do Desenvolvimento Ágil, que, principalmente, busca retomar as origens da engenharia de software e das boas práticas de programação.

Segundo *Robert* relata, no Código Limpo, no início da programação de computadores (década de 60), os desenvolvedores eram matemáticos e pessoas envolvidas com ciência, cálculos, etc. E, alguns princípios e boas práticas estavam "no sangue" em seu trabalho.

Com o passar do tempo, a demanda cresceu e a quantidade de pessoas tornando-se programadores, sem o background científico, também cresceu. Com isso, houve uma perda da cultura inicial e o que se viu foi uma "correria" para escrever códigos cada vez mais difíceis de entender e manter.

Background histórico

Ainda hoje, é bastante comum memes e piadas do tipo "*Quando eu escrevi esse código, só Deus e eu sabíamos entender. Hoje, só Deus.*". Esse tipo de software, além de irresponsável, pode custar caro à empresa, até sua ruína.

O Movimento Ágil surgiu com o objetivo de alinhar velocidade e responsabilidade ao desenvolvimento de software.

Um pouco mais de cada princípios

Vamos falar sobre o que cada um desses princípios busca nos trazer:

- Single-Responsibility Principle: Esse princípio nos diz que uma classe deve possuir apenas uma responsabilidade.
- Open-Closed Principle: Entidades de software devem estar abertos para extensão, mas fechados para modificação.
- Liskov Substitution Principle: Objetos de uma superclasse podem ser substituídos por qualquer de seus subtipos sem alteração do comportamento.
- Interface Segregation Principle: Nenhuma classe deve depender de métodos que ela não usa.
- Dependency Inversion Principle: Dependências de código-fonte se referem apenas a abstrações e não a itens concretos.

Outros princípios

Além do SOLID existem outros conjuntos de princípios com o mesmo objetivo: *manter um código capaz de ser entendido, mantido e expandido.*

- DRY (Don't-Repeat-Yourself): Não repita código. Use abstrações e heranças. Cada mudança, não deve provocar efeitos cascatas ou danos colaterais. O princípio está descrito no livro "O programador pragmático".
- YAGNI (You-Ain't-Gonna-Need-It): Princípio criado na eXtreme Programming (XP), onde uma funcionalidade ou trecho de código só deve ser escrito quando necessário, nunca "para o futuro".
- KISS (Keep-It-Simple-Stupid): Acrônimo criado na *US Navy* nos anos 60, define a simplicidade como objetivo máximo no design de soluções.

Obrigada